Appendix

1. Appendix

The architecture of Self-attention Decoder is shown in Figure 1. Since the Self-attention Decoder architecture is permutation-invariant, we could shuffle the order of the vector and get the same result. For each vector $I_i([1:k])$ in feature sequence, we embed the index i into position vector E_i with the same dimension as the feature vector I_i by an embedding operation defined below:

$$\begin{cases} E_{(index,2x)} = \sin(index/10000^{2x/c}) \\ E_{(index,2x+1)} = \cos(index/10000^{2x/c}) \end{cases} \tag{1}$$

where $index$ denotes the position of the feature vector in the feature sequence, c refers to the dimensions of E, which is the same as that of feature vector, and $2x$, $2x+1$ denote even and odd dimension, respectively, i.e., the sin value is added to the even dimension and the cos value is added to the odd dimension of the feature vector. After that, the position-embedding feature E and the flatted input feature I are added to obtain the fused feature F, which is location-sensitive.

We construct four decoder layers in series to decode the fusion feature F and obtain the character sequence. Each decoder layer consists of two multi-head attention modules, i.e. a feed-forward module, where the last decoder layer has an additional prediction module, and the multi-head attention module connected by four self-attention modules. As shown in Figure 1, each self-attention module includes *query(Q)*, *keys(K)* and *values(V)s*, which are obtained by linear transformation of the self-attention module's input. The equations are shown below:

$$\begin{cases} Q = W_Q \times X \\ K = W_K \times X \\ V = W_V \times X \end{cases} \tag{2}$$

where W_Q, W_K and W_V are the linear transformation matrices corresponding to *Q*, *K*, and *V*, respectively; X refers to the input of self-attention module. Given a query vector Q and a key vector set K, attention mechanism computes the similarity between the query q and all keys K_i, and then aggregating the values V_i based on the similarity. So we can

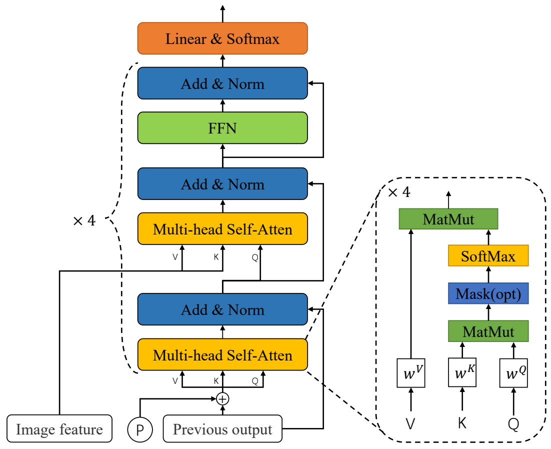

Figure 1. The overview of the Self-Attention module.

calculate the output of Self-Attention as follows:

$$Attention(Q,K,V) = Softmax(\frac{Q \times K^T}{\sqrt{d_K}}) \times V \tag{3}$$

here, d_K is the dimensions of the column for Q and K, which is used to prevent the inner product of Q and K from being too large.

The output of multi-head attention modules is obtained by connection the result of several self-attention modules together, and the *Q*, *K*, *V* of each self-attention modules is calculated by a different linear transformations. This multi-head attention modules can be summarized as:

$$\begin{cases} MultiHead(Q,K,V) = Concat(H_1,H_2,H_3,H_4) \\ H_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \end{cases} \tag{4}$$

Where ,W_i means different trainable linear transformation matrices for each self-attention module.

The recognition process follows the order of the characters in the word, i.e., the $(i+1)$-th character could be recognized only after the i-th character is recognized. To address this issue, we add a Masked operation to the first multi-head attention module in Self-attention Decoder, which can prevent the i-th character from knowing the information after the $(i+1)$-th character.

The second multi-head attention module is different from the first one in that its K and V are computed from the fused feature F. In this way, the prediction of each character is able to use the whole feature extracted by CNN Encoder.

We introduce the feed-forward module, which is made up of two linear layers and one ReLU layer, to the Self-attention Decoder in order to provide the network with non-linear transformations. After that, in order to speed up the convergence, we attach an add&norm layer on each module where add means a residual connection and norm means Layer Normalization [1]. The first and second multi-head attention module and their corresponding add&norm layer consists of the masked multi-head self-attention module and multi-head attention module, respectively.

At last, we take the output of the last decoder layer as the input of classification module, which consists of a single linear layer and softmax layer.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 2